MaxScale

An Intelligent Data Gateway

© 2015, MariaDB Corp.
© 2015, MariaDB Corp.

Problem

- Database scalability
  - example: Master/slave with 300 slaves
- Disaster recovery
  - same example: Master crash
- Real-time data streaming to OLAP/DW and Big Data stores
- Copy data to other applications, QA databases

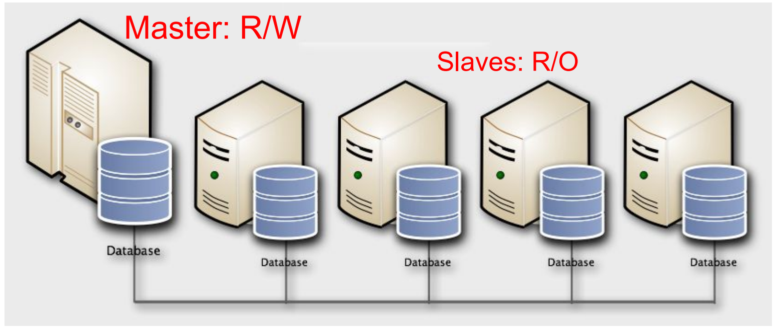
© 2015, MariaDB Corp.

Agenda

- Existing setups
- An Intelligent Data Gateway (IDG) approach
- Setup examples
- Inside Maxscale
- Setup hints and examples
- Binlog router and crash recovery
- Additional features

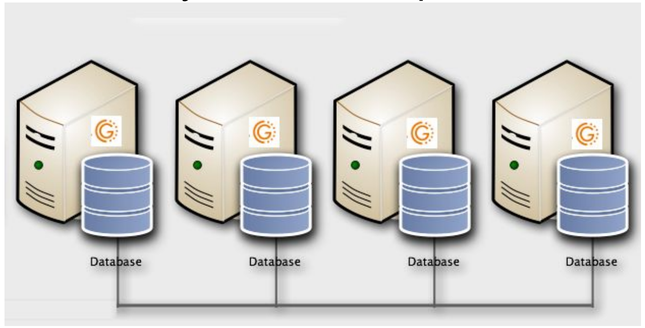
© 2015, MariaDB Corp.

Master/Slave and Galera; HA



- Client have to be read/write aware
- Master crash?

Galera: Synchronous replication



- slow with high rate and big transactions

Pacemaker ? can monitor only machine status, not DB state

© 2015, MariaDB Corp.

The MaxScale Concept: An Intelligent Data Gateway

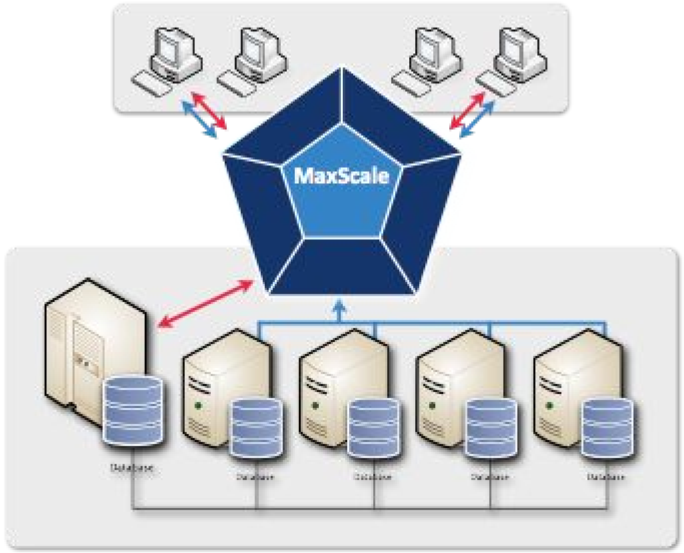
- Decouple applications from database deployment environment
  - Improve availability without adding application complexity
  - Improve data security
  - Handles scale-out issues
  - Add flexibility without burdening every application

- Enable data replication from OLTP databases to external data stores
  - Improve database scalability
  - Remote data disaster recovery
  - Real-time data streaming to OLAP/DW and Big Data stores
  - Copy data to other applications, QA databases

© 2015, MariaDB Corp.

Setup example - improving classic setup

“Replication aware” application: can use 2 connections, one for the master, one for the slave



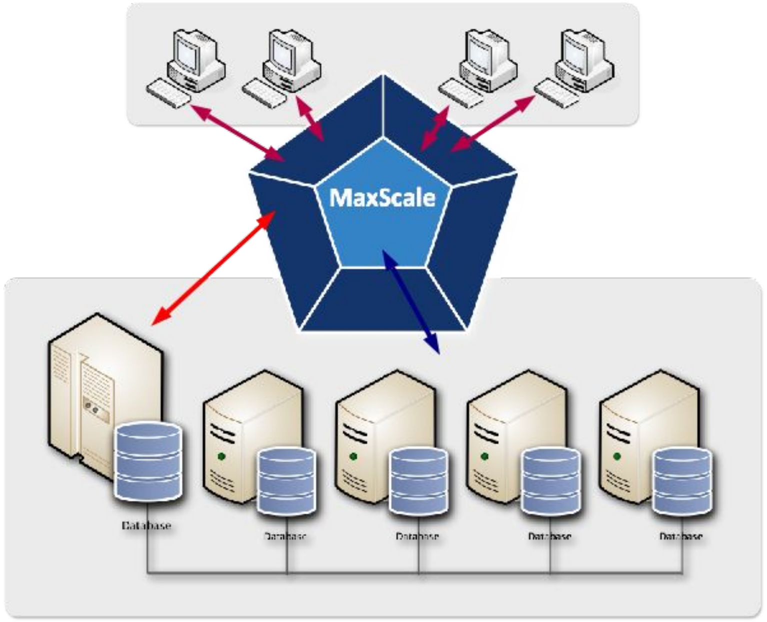
- simple load balancing: fast, lightweight and it provides HA

- MaxScale connects the R/W client connection to the master and it load balances the R/O client connection to one of the slaves

© 2015, MariaDB Corp.

Setup example - improving classic setup

Applications that have been designed to work with a single server and require read scalability



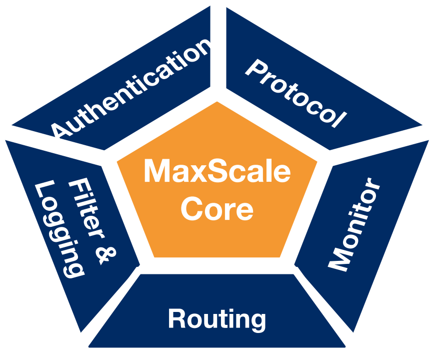
- Each application server uses only 1 connection
- MaxScale monitors the state of each node and it applies load balancing only on the available slaves
- MaxScale creates 2 connections, one for R/W on the master node and one R/O load balanced on the slave nodes

RW Split router

Inside: Pluggable Architecture

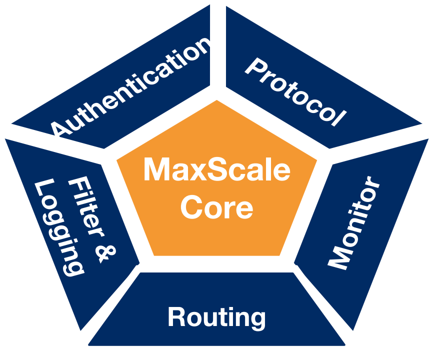
- Generic Core

- Flexible, easy to write plugins for
  - Protocol support
  - Database monitoring
  - Query Transformation and Logging
  - Load balancing and Routing
  - Authentication



© 2015, MariaDB Corp.

MaxScale Core



- Provides core services for
  - configuration
  - networking
  - scheduling
  - query classification
  - logging
  - buffer management
  - plugin loading
  - request flow
- Designed to make plugins easy to write

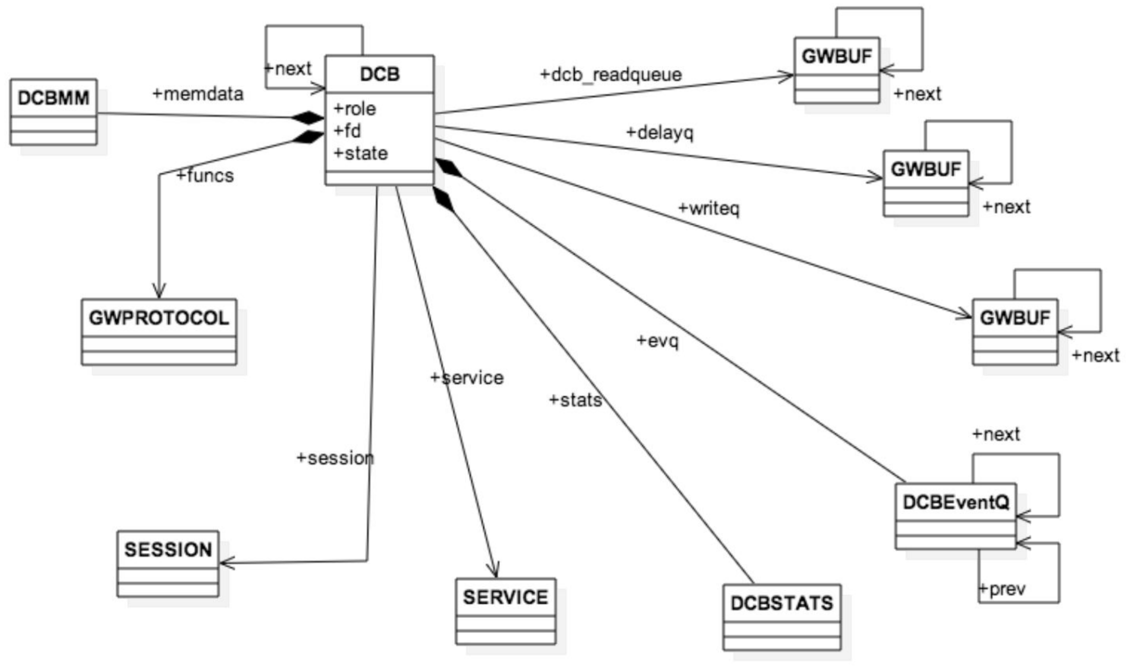
How does it work??

- Event Driven Model
    - All network I/O is event driven
    - Each connection (file descriptor) is given a Descriptor Control Block (DCB)
    - The file descriptor and DCB are registered with EPOLL on linux
    - Every change of descriptor state causes an EPOLL event, DCB is event data
- DCB
    - Connection state must be passed to event handlers
    - DCB is mechanism for communicating connection state between event handlers
    - On arrival events are queued for processing by MaxScale threads
- threads:
    - utility threads - may block
        - log manager
        - housekeeper
        - one per monitor plugin
    - I/O threads
        - all network sockets have to be non-blocking
        - never wait for response on network thread

© 2015, MariaDB Corp.

Descriptor control block (DCB)

- DCB is centre of polling
- All events are passed the DCB
- The DCB holds all the connection state directly or indirectly
- DCB's maintain queues of outgoing and incoming data



DCB - buffer queues

- DCB--->writeq
  - The DCB write queue is used when data is being written to the socket and the socket buffer becomes full.
  - Instead of the write blocking the residual data is added to the DCB writeq
  - The writeq will be flushed when an EPOLL_OUT event is received on the descriptor
- DCB--->delayq
  - The delay queue is used to hold requests when there is an outstanding authentication handshake on the connection
- DCB--->dcb_readqueue
  - The dcb_readqueue buffers incomplete requests

© 2015, MariaDB Corp.

DCB - GWPROTOCOL

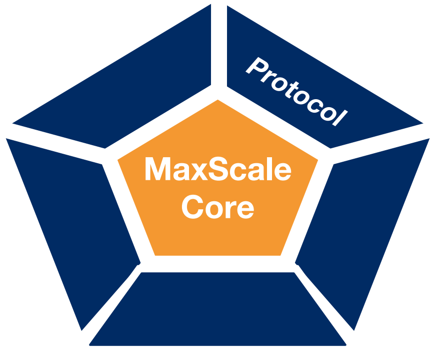
- GWPROTOCOL is the link between the DCB and the protocol plugin
- A set of function pointers that are entry points in the protocol plugin
- These entry points include:
    - read, write, write_ready, error, hangup, accept, connect, close & listen operations
    - These entry points are the links to the protocol specific part of the event handler

DCB - Memory Management

- DCB's may be referenced from multiple I/O threads
- A DCB can not be freed until all threads have finished with it
- DCBMM manages:
    - A bitmap - one bit per thread
    - A zombie list
- A DCB is first placed on zombie list
- As each thread completes event processing it checks the zombie list and clears it's bit
- Only when all bits are cleared the DCB is freed

© 2015, MariaDB Corp.

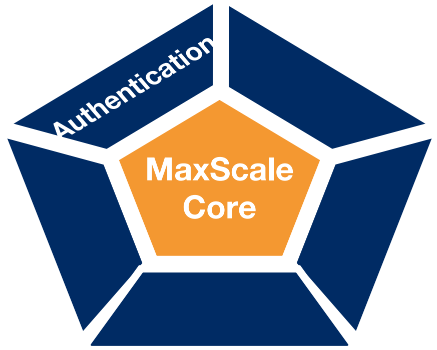
Protocol Plugin



- Provides for protocol implementations
  - Client to MaxScale
  - MaxScale to Database
- Also used for administration protocols
- Potential to allow for non-MariaDB database protocols

© 2015, MariaDB Corp.

Authentication Plugin

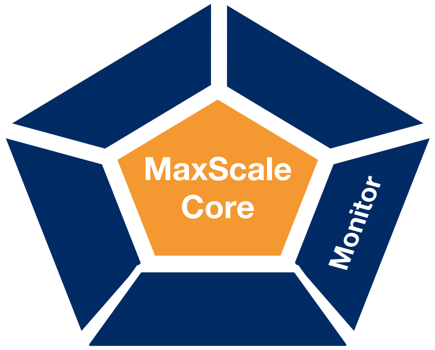


- Provides for authentication mechanisms
  - Client to MaxScale
  - MaxScale to database
- Responsible for mapping disjoint authentication schemes

© 2015, MariaDB Corp.

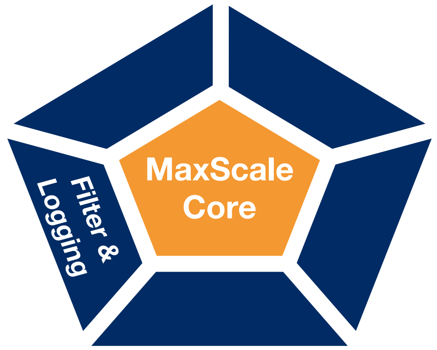
Monitoring Plugin

- Monitors the database environment
- Provides availability and status information
- Tailored to particular database configurations



© 2015, MariaDB Corp.

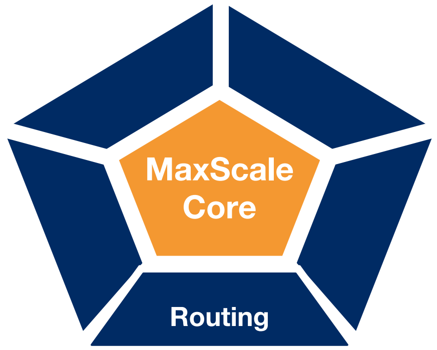
Filter Plugin



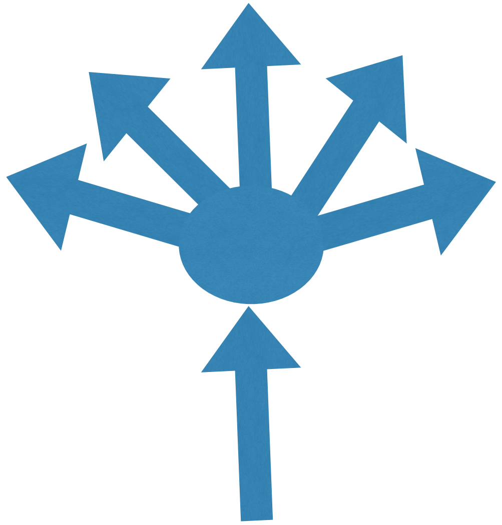
- A filter may modify, block or log a request as it passes through MaxScale
- Filters may be built up into chains
- Filters may duplicate requests

© 2015, MariaDB Corp.

Router Plugin

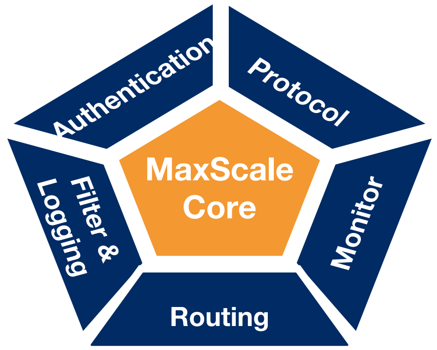


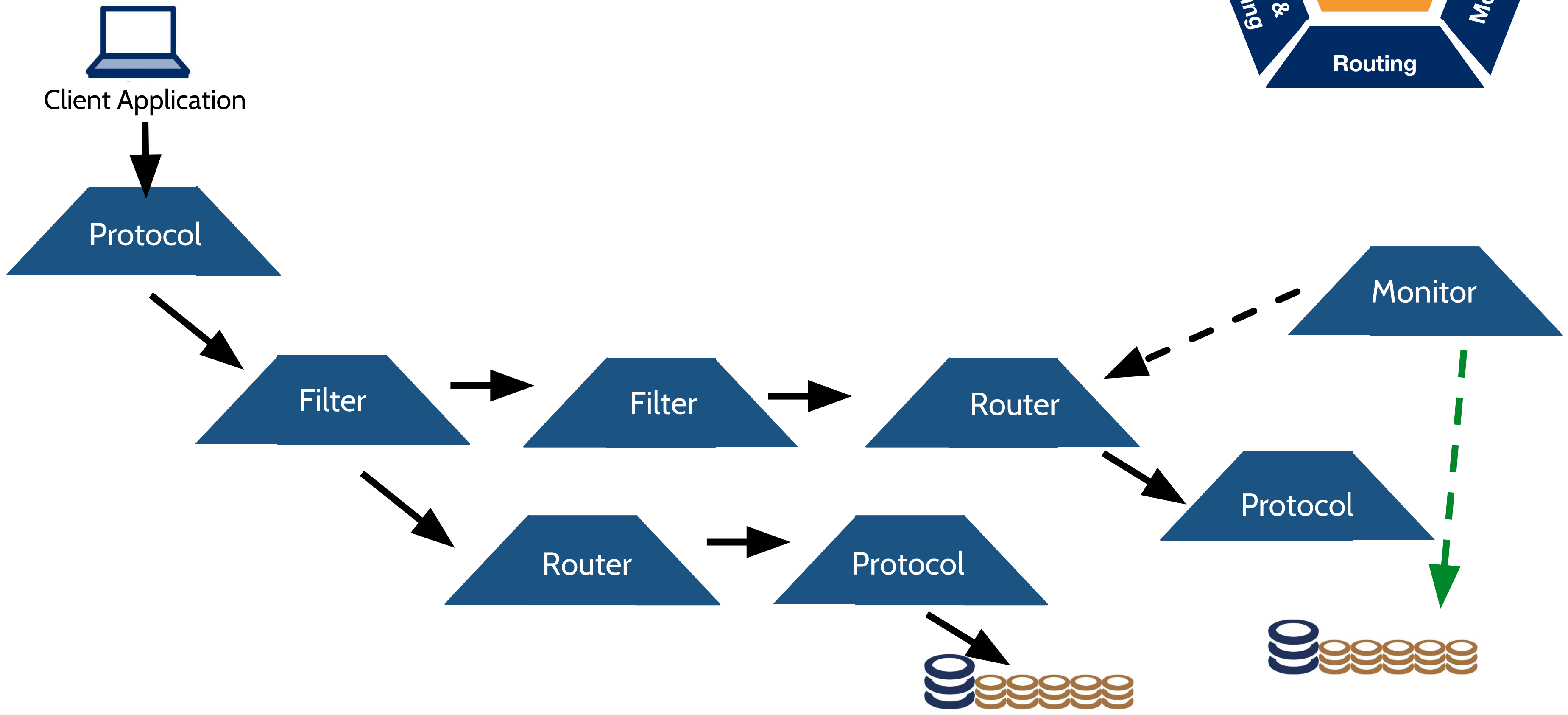
- Routes requests to backend database servers using a combination of
  - Data from monitoring
  - Routing algorithm
  - Hints from filters
  - Request characteristics
- Classes of router
  - Connection routing
  - Statement routing



© 2015, MariaDB Corp.

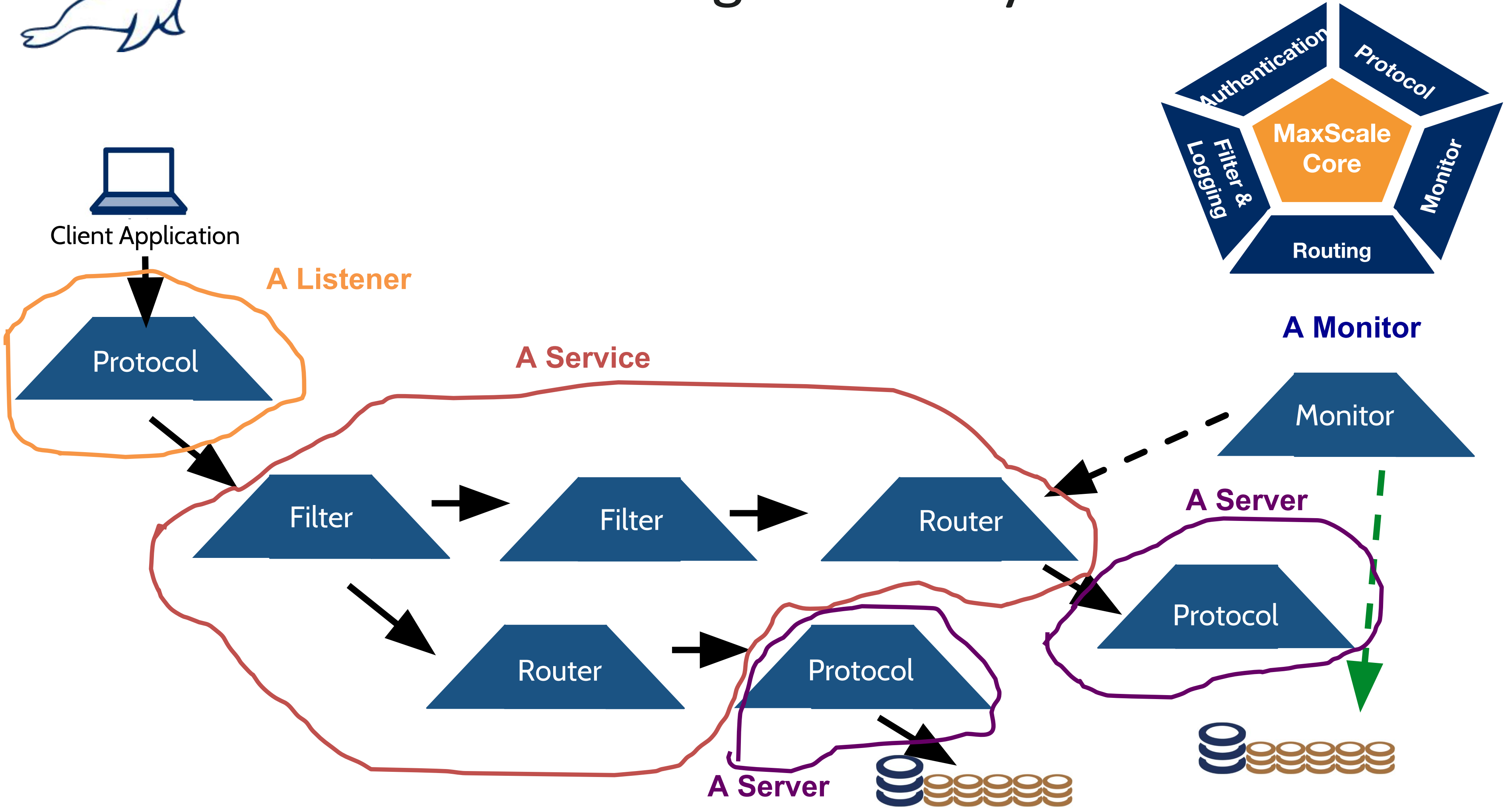
All together





© 2015, MariaDB Corp.

Maxscale config: Anatomy of a service



© 2015, MariaDB Corp.

Maxscale config: General Section

- General configuration options in the maxscale section of ini file
- Number of threads
- Enable or disable log levels

```
[maxscale]
threads=3
log_messages=off
log_debug=on
```

© 2015, MariaDB Corp.

Bold – Mandatory
Regular-optional

End to End Configuration for a Service

```
[ListenerNAme]
type=listener
service=<ServiceName>
protocol=MySQLClient
port=<MaxScale Port for listening>
```

```
[BackEndserverName]
type=server
address=<server-host-address>
port=<port-at-which-database-server-listens>
protocol=MySQLBackend
```

One for each server in cluster

```
[<ServiceName>]
type=service
router=<router-module-name>
router_options=<router-options specific to router>

servers=<list of backend servers to route to srv1, srv2>
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
<service-specific-option>=<option-value>
Filter=<FilterName>
```

```
[FilterName]
type=filter
module=<filter-module-name>
<filter-specific-option>=<option-value>
```

```
[MonitorName]
type=monitor
module=<monitor-module-name>
servers=<list of servers in cluster to be monitored srv1, srv2>
user=mper
passwd=massi
monitor_interval=<value>
```

Master-Slave Cluster Read & Write Services Configuration

- Read Service

```
[ReadService]
type=service
router=readconnroute
router_options=slave
servers=server1,server2,server3,server4
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

- And network listener

```
[ReadListener]
type=listener
service=ReadService
protocol=MySQLClient
port=4006
```

- Write Service

Note the router_option between read and write service

```
[WriteService]
type=service
router=readconnroute
router_options=master
servers=server1,server2,server3,server4
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

- And network listener

```
[WriteListener]
type=listener
service=WriteService
protocol=MySQLClient
port=4007
```

Remember to configure the client applications to use these ports to send the queries to MaxScale

© 2015, MariaDB Corp.

Define the servers

```
[ReadService]
type=service
router=readconnroute
router_options=slave
servers=server1,server2,server3,server4
user=mper
passwd=6628C50E07CCE1F0392EDEEB9D1203F3
```

- Define the address/hostname
- The port on the server at which MySQL is listening
- The protocol module to use for connection

```
[server1]
type=server
address=127.0.0.1
port=3307
protocol=MySQLBackend
```

Configure rest of the servers (server2,server3, server4) as well

- Optionally override the monitor user and password for this server

MySQL(master-slave) Monitor Configuration

- Single Monitor for both services

- Use mysqlmon module

```
[MySQL Monitor]
type=monitor
module=mysqlmon
servers=server1,server2,server3,server4
user=mper
passwd=massi
```

A Group of Server to monitor as a cluster

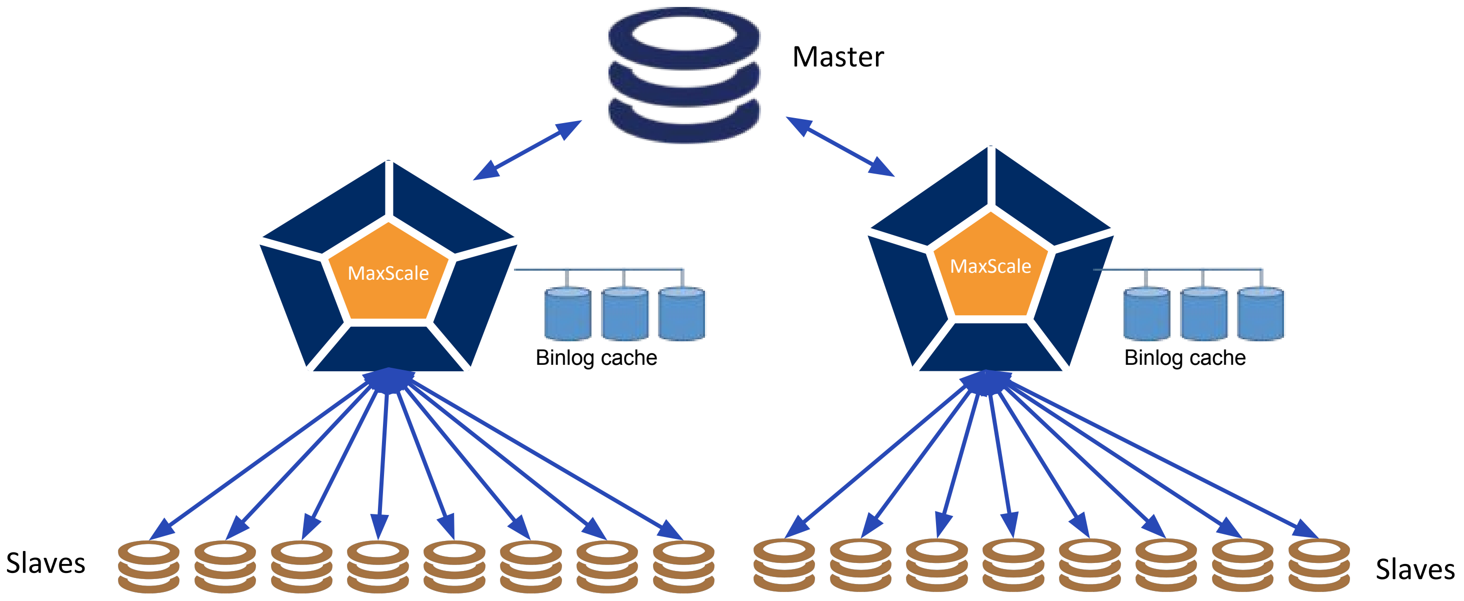
- Optionally define monitoring interval (in milliseconds)

```
monitor_interval=5000
```

- Reducing the monitor interval allows to detect failure faster

More example: Binlog router

- Transparent MariaDB binlog replication relay

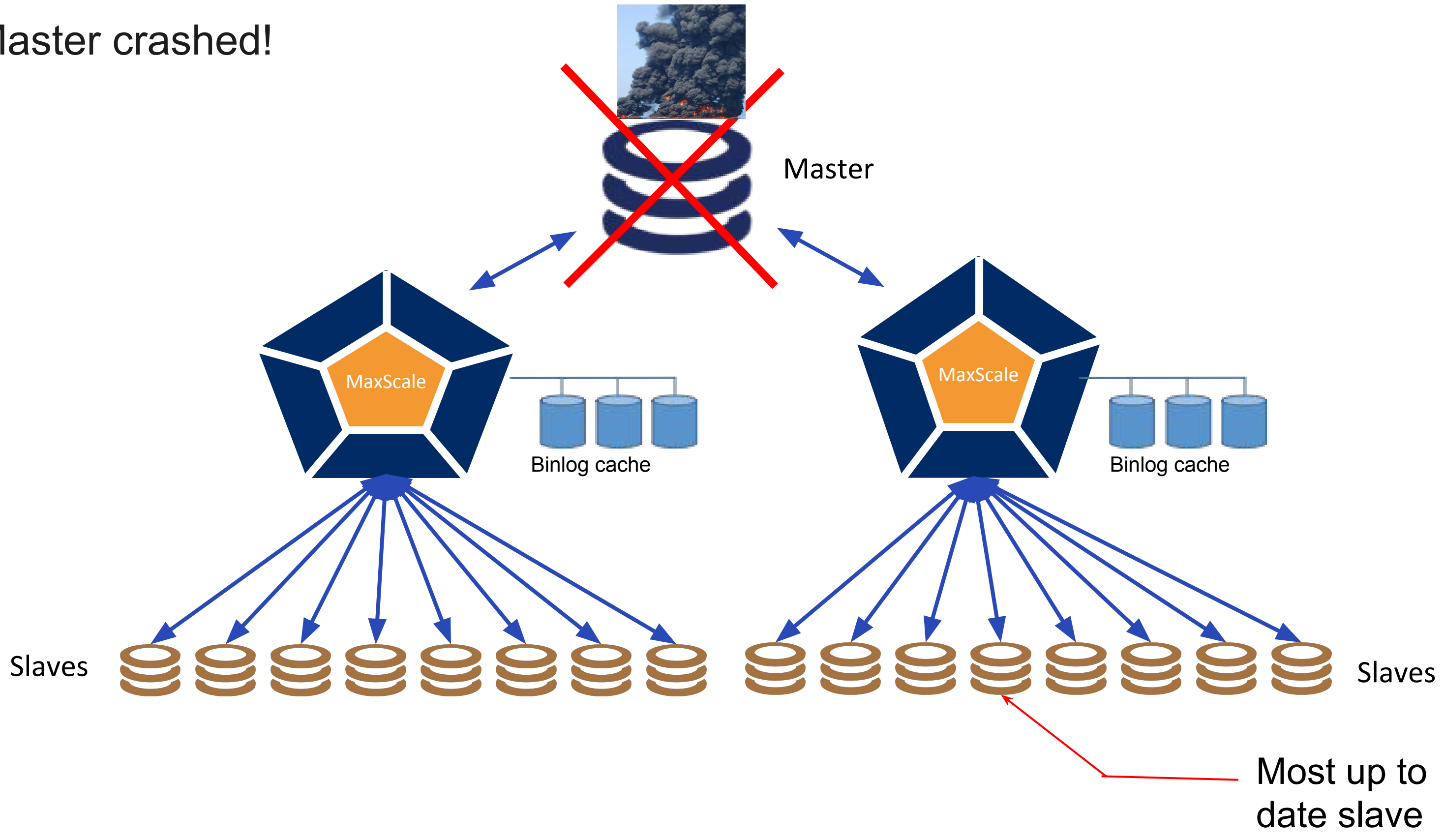


- Horizontally Scale Slaves without Master Overload
- Better Parallel Replication

© 2015, MariaDB Corp.

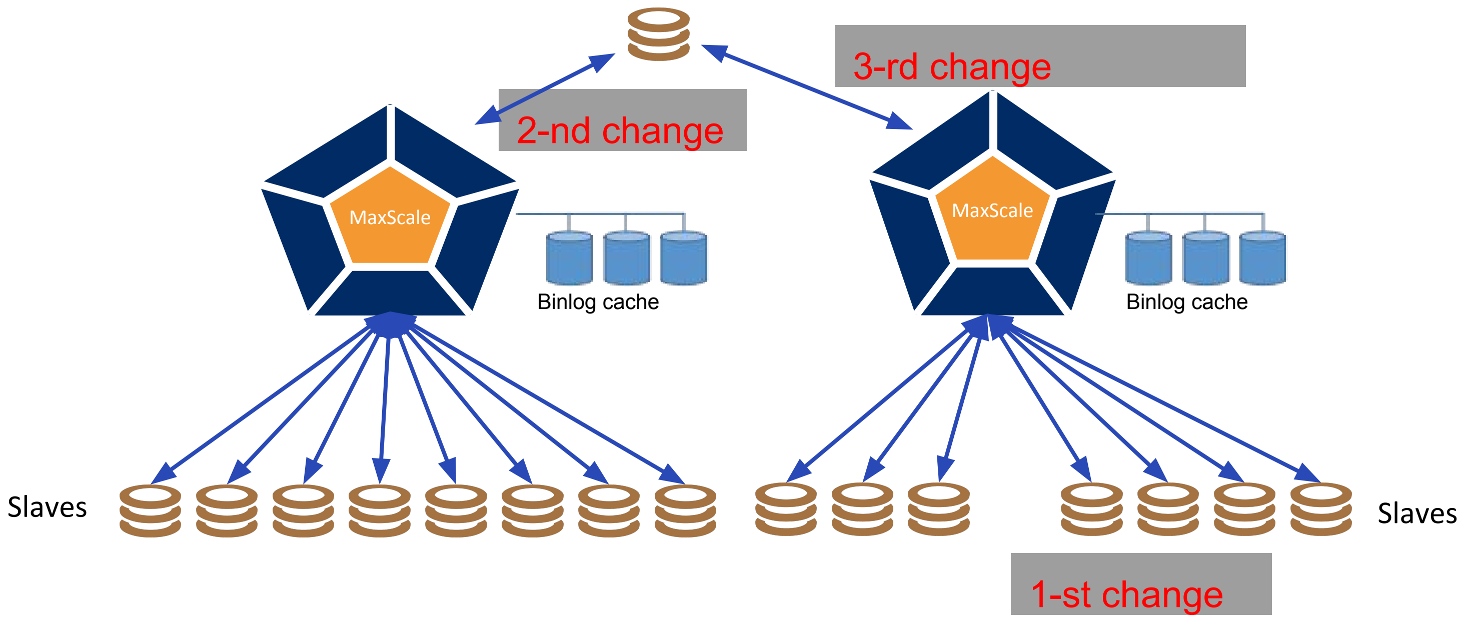
Binlog router: crash recovery

- Master crashed!



Binlog router: crash recovery

- No need to touch hundreds slaves



© 2015, MariaDB Corp.

Example Configuration for Binlog Router

```
[Binlog_Service]
type=service
router=binlogrouter
servers=master
user=repl
passwd=slavepass
version_string=5.6.15-log
router_options=server-id=313,binlogdir=/servers/binlogs,filestem=mysql-bin

[master]
type=server
address=master.example.com
port=3306
protocol=MySQLBackend

[Binlog Listener]
type=listener
service=Binlog_Service
protocol=MySQLClient
port=5306
```

Port on Master Server to which MaxScale will connect to

Port on MaxScale where Slave will connect to

Command on Slave to use MaxScale as Master

- CHANGE MASTER TO MASTER_HOST=<MaxScale-Host-name> MASTER_PORT=5306, MASTER_LOG_FILE='mysql-bin.00001'

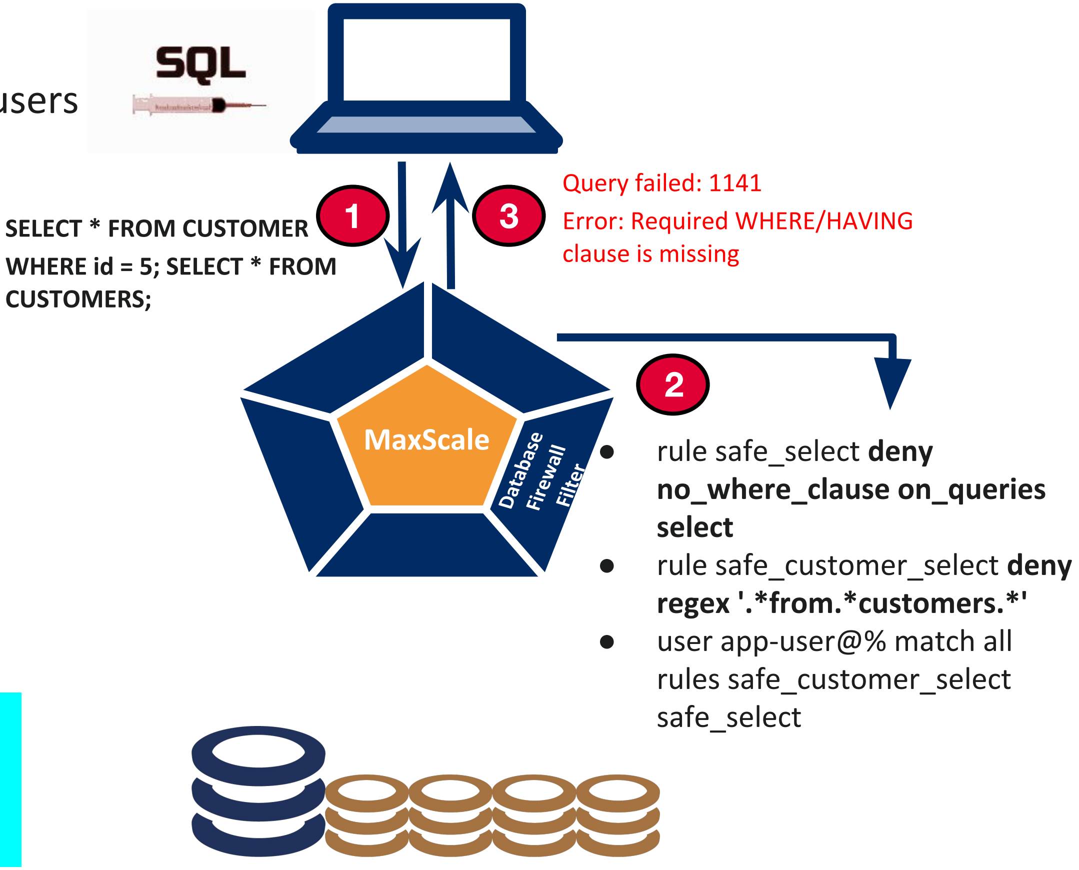
© 2015, MariaDB Corp.

Security: Query Blocking

- Block queries that match a set of rules
- Block queries matching rules for specified users
- Multiple ordered rules

- Match on and block queries with certain patterns
  - date/time
  - a WHERE clause
  - Query type
  - Column match
  - a wildcard or regular expression

- Protect against SQL injection
- Prevent unauthorized data access
- Prevent data damage


SQL
1
SELECT * FROM CUSTOMER WHERE id = 5; SELECT * FROM CUSTOMERS;
3
Query failed: 1141
Error: Required WHERE/HAVING clause is missing
2
MaxScale
Database Firewall Filter

- rule safe_select **deny no_where_clause on_queries select**
- rule safe_customer_select **deny regex '.*from.*customers.*'**
- user app-user@% match all rules safe_customer_select safe_select

Security: Query Blocking

Example configuration

- Rules are defined in a separate file

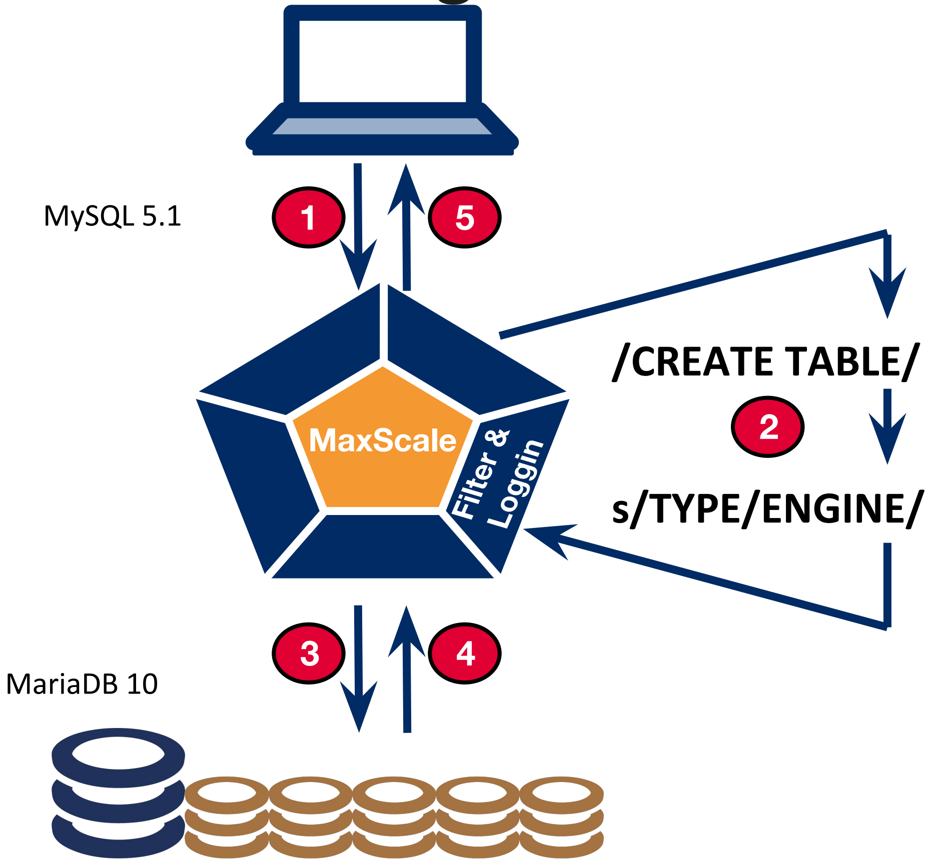
```
[Connection Service1]
type=service
router=readconnroute
servers=server1
user=maxuser
passwd=maxpwd
filters=Firewall
```

```
[Firewall]
type=filter
module=dbfwfilter
rules=/home/user/myrules.txt
```

```
rule safe_select deny no_where_clause on_queries select
rule safe_customer_select deny regex '.*from.*customers.*'
user app-user@% match all rules safe_customer_select safe_select
```

© 2015, MariaDB Corp.

Migration: Query Transformation - regex filter



Modify queries from legacy applications on the fly - for example a MySQL 5.1 app:

1. MaxScale accepts a query from a MySQL 5.1 compatible client,
2. If the query matches the regular expression "/CREATE TABLE/" then MaxScale substitutes "ENGINE" for "TYPE" in that statement, else it passes the statement through the filter unchanged.
3. Forwards the transformed statement to MariaDB 10.0
4. Receives the result from the back-end.
5. Forwards the result to the client.

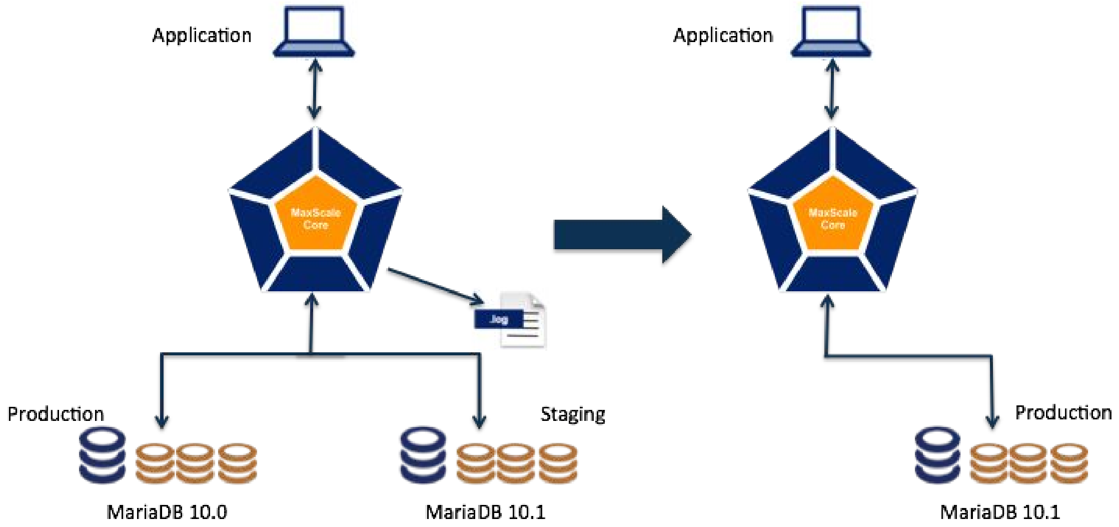
- Application and Database can be upgraded asynchronously
- Faster rollout of new database versions

Acts like linux 'sed'

© 2015, MariaDB Corp.

Migration: Upgrade from one version to another version - QLA filter

- Tee-filter to duplicate queries to
  - current version in production
  - new version in staging
- QLA filter logs query performance
  - Queries sent to new version
  - Validate Performance
- QLA filter logs query syntax
  - Queries sent to new version
  - Validate functionality

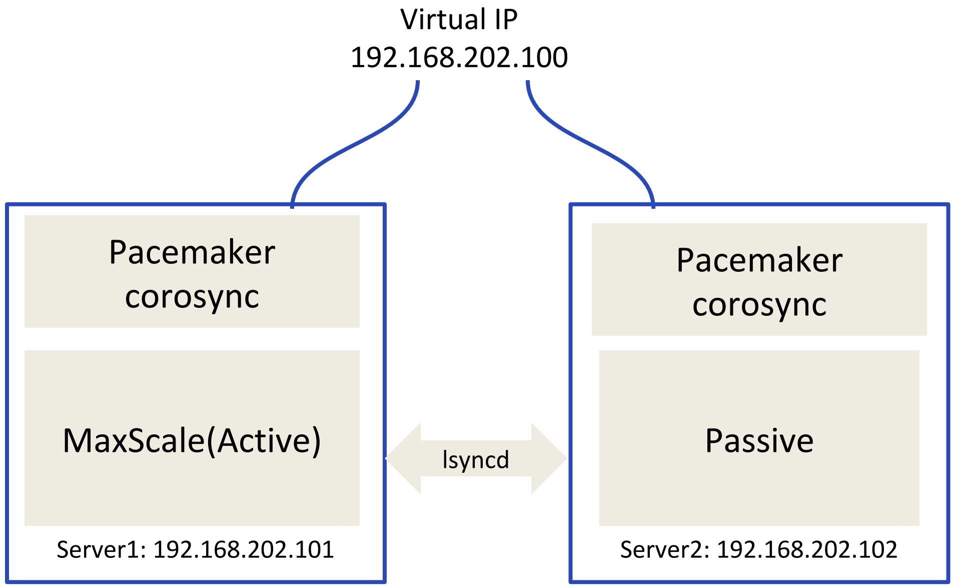


- Validate functionality and performance on new version before moving to production
- Minimize risk

© 2015, MariaDB Corp.

MaxScale HA using pacemaker and lsyncd

- Virtual IP
- Cluster Management: Pacemaker/corosync
  - monitor
  - start
  - stop
  - restart
  - move
- Pacemaker/corosync configuration example
  - https://github.com/mariadb-corporation/MaxScale/blob/develop/Documentation/Reference/MaxScale-HA-with-Corosync-Pacemaker.md
- MaxScale.cnf synchronization: lsyncd
  - By default lsyncd will search for the configuration file in /etc/lsyncd.conf



© 2015, MariaDB Corp.

Get involved!

- Source is open: https://github.com/mariadb-corporation/MaxScale/
  - query classified is based on MariaDB code https://github.com/MariaDB/server
- Road maps and Bugs: https://mariadb.atlassian.net/browse/MXS
- Blogs: https://mariadb.com/blog-tags/MaxScale
- Discussion via the Google Group: maxscale@googlegroups.com
- Knowledge base https://mariadb.com/kb/en/mariadb-enterprise/mariadb-maxscale/

© 2015, MariaDB Corp.